《C/C++学习指南》

第19.2讲：面向对象编程的设计思想-2

作者：邵发　　QQ群：417024631
官网：http://www.afanihao.cn/
答疑：http://www.afanihao.cn/kbase/

实例：DataStore

第二步：细化为函数

定义一个类型

```
struct DataStore
{
};
```

(注：成员变量先不用考虑)

创建与销毁

```
DataStore*  ds_create();
void ds_destroy(DataStore*  store);
```

实例：DataStore

第二步：细化为函数

其他功能函数：

(1) 可以向它加入一个记录

```
void ds_add( DataStore* store, const Student* obj);
```

(2) 可以按ID来查找一个记录

```
Student* ds_find(DataStore* store, int id);
```

(3) 可以按ID删除一个记录

```
void ds_remove(DataStore* store, int id);
```

(4) 可以打印显示所有的记录

```
void show_all(DataStore* store);
```


实例：DataStore

第二步：细化为函数

确定该对象的使用方式

```
// 创建一个对象
DataStore* store = ds_create();

// 调用ds_add
Student obj;
ds_add(store, &obj);

// 销毁对象
ds_destroy(store);
```

第三步：选择一种实现

使用链表来实现

把相关的数据放在DataStore对象里

```
struct DataStore
{
      Student head; // 存一个有头链表
};
```

C/C++学习指南 邵发 www.afanihao.cn

第三步：选择一种实现

创建对象

```
DataStore*  ds_create()
{
      // 动态创建对象
     DataStore* store = (DataStore*)malloc(sizeof(DataStore));

      // 初始化
      store->head.next = NULL;
      return store;
}
```

第三步：选择一种实现

销毁对象

```
void ds_destroy(DataStore*  store)
{
	// 释放所有相关资源
	Student* p = store->head.next;
	while(p)
	{
		Student* next = p->next;
		free(p);
		p = next;
	}
	// 销毁对象
	free(store);
}
```


第三步：选择一种实现

添加对象

```
void ds_add( DataStore* store, const Student* obj)
{
	// 创建对象、复制数据
	Student* copy = (Student*)malloc(sizeof(Student));
	*copy = *obj;

	// 添加到链表
}
```

第三步：选择一种实现

添加对象

```
void ds_add( DataStore* store, const Student* obj)
{
	// 创建对象、复制数据
	Student* copy = (Student*)malloc(sizeof(Student));
	*copy = *obj;

	// 添加到链表
}
```


.h / .cpp分离

调用者不关心函数的实现。。。

分析

1. DataStore是一种类型，用户可以创建多个DataStore，来存储不同的数据。每个对象之间互不影响。

2. 使用者只关心如何使用，并不关心内部的实现。